

-1-

Date: <u>May 31, 2001</u> Express Mail Label No. <u>EL 552284505 us</u>

Inventor: Thomas W. Nickerson
Attorney's Docket No.: 1958.2010-000

SYSTEM AND METHOD FOR DISPLAYING DYNAMIC PAGE CONTENT IN A PAGE-CACHING BROWSER

BACKGROUND

Client-side browsing applications, such as web browsers, are tools for viewing
5 multimedia content, such as web pages. A web page is displayed by loading one or
more script files that define page content and function into a browser. According to
various embodiments, script files may be encoded in a markup language, such as HTML
or XML, and may also include instructions from scripting languages, such as JavaScript.
Script files may reference other script files and/or content files, which store formatted
10 graphics, audio and/or video for additional display and sound.

Web pages are typically served upon request by a host server. A web browser
requests a web page from the server by specifying an address, such as a URL (Universal
Resource Locator), in a content request message, such as an HTTP request message
(*i.e.*, HyperText Transport Protocol). A URL is an address identifying the location of a
15 resource, such as a script file or content file. User-specified parameters may be
appended to the URL in the form of a query string (*i.e.*, “?<parameter1>=<value1>...”)
providing for customized and/or interactive page content implemented through
server-side processing.

Most browsers have the ability to locally cache web pages by caching the
20 constituent script and content files on the client computer's hard drive as they are
loaded. Caching improves browser performance such that a web page can be loaded
directly from cache memory, avoiding additional network and processing delays.

SUMMARY

Embodiments of the invention include a system and method for displaying dynamic page content in a page-caching browser, which avoids the display of unintended page content in a browser. Such embodiments include (i) specifying an address to stored content; (ii) appending a unique identifier to the address; (iii) 5 requesting the content with the address and the appended identifier; and (iv) transmitting the content request to the server regardless of whether there is cached content associated with the address.

According to various embodiments, the address includes a Universal Resource Locator (URL) to content of at least a portion of a web page and a query string in which 10 a unique identifier is appended to the address.

The unique identifier may be generated in a number of ways known to those skilled in the art. For example, in one embodiment, the unique identifier may be a random number generated by a script function, such as the Math.random() method 15 defined in JavaScript. In another embodiment, the unique identifier may be a system time stamp. In a further embodiment, the unique identifier may include any unique alpha-numeric or other symbolic representation.

Embodiments of the invention may be implemented in a system in which a client, such as a web browser, is coupled to a server to access state and function for a 20 user session. The client specifies an address to content stored on the server and appends a unique identifier to the address. The client requests the content with the address and the appended identifier, which is transmitted in a content request to the server regardless of whether there is cached content associated with the address to the content.

Embodiments of this invention make page navigation for stateful and 25 dynamically changing web pages completely translucent to the user. Such embodiments cause the browser to ask the server for a new page when new information is required regardless of whether the browser is configured to always load content from cache.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of embodiments of the invention, as illustrated in the accompanying drawings in which like reference
5 characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

FIG. 1A is a diagram illustrating a typical web browsing system that implements caching.

10 FIG. 1B is a table illustrating how loading from cache can result in a stateful web page displaying unintended page content.

FIGS. 1C through 1G illustrate various states of a web page displaying a car having selectable exterior and interior colors according to FIG. 1B

15 FIG. 2A is a diagram illustrating a web browsing system that requests content from a server regardless of whether caching is enabled according to one embodiment.

FIG. 2B is a table illustrating how unintended page content is avoided regardless of whether caching is enabled according to one embodiment.

FIGS. 2C through 2G illustrate various states of a web page displaying a car having selectable exterior and interior colors according to FIG. 2B

20 FIG. 3 is a diagram illustrating how the display of unintended page content is avoided in response to redirection by a server according to one embodiment.

FIG. 4 is a flow diagram illustrating a process for displaying dynamic or stateful page content in a page-caching browser according to one embodiment.

DETAILED DESCRIPTION

25 FIG. 1A is a diagram illustrating a typical web browsing system that implements caching. The system includes a client computer 100 executing a web browsing application 110, which is capable of storing and loading web pages to and from a local cache 120. A web site 200 includes a host server 210 that processes content requests

from the browser 110 and responds with the requested script or content files. The host server 210 may have access to a cache 220, such as a database, for storing such files.

Typically, a browser can be configured with different options that specify the frequency for loading page content from a local cache. Typical options include

- 5 attempting to load from cache (i) every time page content is requested, (ii) never, or (iii) one or more intermediate criteria. When a script or content file of a web page is cached, the cached file is associated with the address specified in the content request. For example, the address may be the URL, and query string if any, used to request and generate the cached script file.

- 10 Referring to FIG. 1A, suppose that a user attempts to load a web page that corresponds to a script file on the server 210 having a URL,

“http://www.mysite.com/PAGE1.HTML”. (1)

- Assuming that the web page has not been previously cached, the browser 110 transmits an HTTP request (reference 300) over the network, such as the Internet. The HTTP
15 request is received and processed by the host server 210, which retrieves the script file, “PAGE1.HTML,” from cache memory 220 (reference 301). The script file is then transmitted back to the browser 110 where it is loaded and displayed to the user (reference 302). With caching enabled in the browser 110, the loaded script file is stored in the local cache 120 and associated with URL (1). (reference 303).

- 20 Depending on the configuration of the browser 110, subsequent requests for URL (1) may cause PAGE1.HTML to be loaded directly from the local cache 120. For example, the browser 110 may be configured to always load content from the local cache 120 when a requested URL/query string address corresponds to cached content. Thus, with respect to FIG. 1A, the browser 110 will load the script file, PAGE1.HTML,
25 from the local cache 120 whenever URL (1) is requested (references 304 and 305).

If a web page is static (e.g., does not change often), loading page content from a local cache improves browser performance by avoiding additional network and processing delays. However, if a web page is updated often or displays content that

depends on the state of the user session, loading page content directly from cache can result in the display of unintended page content.

FIG. 1B is a table illustrating how loading from cache can result in a stateful web page displaying unintended page content. For example, assume a web page displays a car having selectable exterior and interior colors, as illustrated in FIGS. 1C through 1G. In its intended operation, a user selects either an exterior color (*i.e.*, white, red, and black) or an interior color (*i.e.*, grey and black). In response, the browser generates a content request specifying an address that includes the URL of the web page and a query string indicating the selected interior or exterior color. The web page is stateful, because the resulting page content is dependent on the previous page content. In other words, if an exterior color is selected, the interior color for the resulting page is derived from the interior color of the previous page content. Likewise, if an interior color is selected, the exterior color for the resulting page is derived from the exterior color of the previous page content as well. The state of a web page is typically maintained at the server for a particular user session.

Referring back to FIG. 1B, the table 400 includes three columns specifying, respectively, the address requested (C1), whether page content is loaded from local cache or host server (C2), and a description of the resulting page content (C3). Each row (R1 through R5) corresponds to a new request.

In table cell (R1, C1) the address requested is,

`http://www.mysite.com/PAGE1.HTML` (1).

If no cached content is locally available for this page, an HTTP request specifying the URL (1) is generated and transmitted over the network to the host server 210. The server 210 processes the request and returns the script file, PAGE1.HTML, to the browser 110. As shown in FIG. 1C, the page content resulting from the initial content request is a car 405 having a “black” exterior color and a “grey” interior color, as depicted in the car seat 410. The script file corresponding to this display, PAGE1.HTML, is stored in a local cache 110 where it is available to be loaded when URL (1) is requested again.

In FIG. 1C, the user clicks on an exterior color palette 420 in the display with a pointing device, such as a mouse, to change the exterior color of the car 405 to “red”.

Referring back to FIG. 1B, the click event triggers an event handler, defined in PAGE1.HTML, generating a new content request to display the car with a “red”

5 exterior. In table cell (R2,C1), the requested address is,

`http://www.mysite.com/PAGE1.HTML?EXT_COLOR="RED"` (2)

The requested address includes an appended query string parameter specifying the “red” exterior color (*i.e.*, ?EXT_COLOR=“RED”). Again, there is no cached content associated with the requested URL/query string address. Thus, an HTTP request is

10 generated and transmitted over the network to the host server 210 specifying the URL/query string (2). The server 210 responds with a modified script file, PAGE1.HTML for displaying the car 405 with “red” exterior and “grey” interior colors, as shown in FIG. 1D. The “grey” interior color is derived from the previous state of the web page (*i.e.*, FIG. 1C) with the state being maintained at the server. The modified
15 script file, PAGE1.HTML, is stored in the local cache where it is available to be loaded when URL/query string (2) is requested again.

In FIG. 1D the user clicks on an interior color palette 430 in the display to change the interior color of the car 405 to “black.” Referring back to FIG. 1B, the click event triggers an event handler, defined in PAGE1.HTML, generating a new content
20 request to display the car 405 with a “black” interior. In table cell (R3,C1), the requested address is,

`http://www.mysite.com/PAGE1.HTML?INT_COLOR="BLACK"` (3)

The requested address includes an appended query string parameter specifying the “black” interior color (*i.e.*, ?INT_COLOR=“BLACK”). Again, there is no cached

25 content associated with the requested URL/query string address. Thus, an HTTP request is generated and transmitted over the network to the host server 210 specifying the URL/query string (3). The server 210 responds with a modified script file for displaying the car 405 with “red” exterior and “black” interior colors, as shown in FIG. 1E. The “red” exterior color is derived from the previous state of the web page

(i.e., FIG. 1D). The modified script file, PAGE1.HTML, is again stored in the local cache 120 where it is available to be loaded when URL/query string (3) is requested again.

Assume now that the user closes the browser and relaunches a new session at a later time. Referring to row R4 of FIG. 1B, if URL (1) is requested again, the browser checks the local cache for cached content corresponding to URL (1). Upon locating the corresponding script file in the cache, PAGE1.HTML is loaded into the browser displaying the expected default car with "black" exterior and "grey" interior colors, as shown in FIG. 1F.

Finally, referring back to row R5 of FIG. 1B, when the user triggers a content request for a car having a "black" interior, the browser will again check the local cache, locate the cached script file, PAGE1.HTML, corresponding to URL/query string (4) and load it into the browser. However, loading the cached script file into the browser causes the unintended display of a car having a "RED" exterior with "black" interior, as shown in FIG. 1G. If the browser transmitted the content request to the host server 210, the page content would have been a car with a "black" interior color and a "black" exterior color, which corresponds to the previous state of the exterior color.

Therefore, if a browser is set to never go back to the server when cached content is available, web pages that contain stateful information may display unintended page content. To address this issue, techniques have been implemented in which a website posts a notice to the browser to turn caching off. However, this is a complicated task with some browsers.

Embodiments of the present invention provide a system and method for displaying dynamic page content in a page-caching browser. The system and method prevent the loading of unintended page content from a local cache into a browser by ensuring the uniqueness of the content request. Such embodiments include

- (i) specifying an address to stored content; (ii) appending a unique identifier to the address; (iii) requesting the content with the address and appended identifier; and
- (iv) transmitting the content request to the server regardless of whether there is cached

content associated with the address. By appending a unique identifier to the requested address, each content request is unique, thus, preventing the browser from locating and loading corresponding content from the cache.

FIG. 2A is a diagram illustrating a web browsing system that requests content
 5 from a server regardless of whether caching is enabled according to one embodiment. The system includes a client computer 100 executing a web browsing application 110, which is capable of storing and loading web pages to and from a local cache 120. A web site 200 includes a host server 210 that processes content requests, such as HTTP requests, from the browser 110 and responds with the requested script or content files.
 10 The host server 210 may have access to a cache 220, such as a database, for storing such files. The client and server processes discussed herein are not limited to being executed in different locations. They may, in fact, execute on the same processing device.

According to embodiments of the invention, the browser application 110 submits a request for page content specifying the address of a script or content file
 15 stored on a server 210 along with a unique identifier appended to the address. According to the embodiment illustrated in FIG. 2A, the web browsing application 110 submits a page request by specifying the URL of a web page along with a unique identifier appended to the URL as a query string parameter (reference 300). For example, in the following URL/query string,

20 `http://www.mysite.com/PAGE1.HTML?&ID=0.123127678` (4)
 the unique identifier, "&ID=0.123127678," is appended to the URL in a query string.

The unique identifier may be generated in a number of ways known to those skilled in the art. For example, in one embodiment, the unique identifier may be a random number generated by a script function, such as the Math.random() method
 25 defined in JavaScript. In another embodiment, the unique identifier may be a system time stamp. In a further embodiment, the unique identifier may include any unique alpha-numeric or other symbolic representation.

The host server 210 processes the request to extract the requested script file, PAGE1.HTML, from a cache 220, such as a database (reference 301). In processing the

HTTP request, the server 210 may disregard the unique identifier in the query string. The server 210 then transmits the script file to the browser 110, where it is loaded and displayed (reference 302). If caching is enabled, the browser 110 stores the loaded script file in the local cache 120 (reference 303), where it is available to be loaded directly when URL/query string (4) is requested again.

However, because the query string contains a unique identifier, it is highly unlikely that a URL/query string will be generated that is identical to URL/query string (4). Subsequent requests for web page, "PAGE1.HTML," may have the same URL, but the unique identifier specified in the query string will be different. When the browser 110 checks the local cache 120 for cached content corresponding to a unique address/identifier pair, the unique identifier prevents such association. Thus, the browser 110 is forced to transmit the request to the server 210 (reference 304). For example, referring to FIG. 2A, a subsequent request for "PAGE1.HTML" may specify the following URL/query string,

www.mysite.com/PAGE1.HTML?&ID=1001.1287998 (5)

Since URL/query string (5) is not identical to URL/query string (4), the browser 120 automatically transmits the request over the network to the server 210 (reference 304). Therefore, such embodiments avoid the display of unintended page content for dynamic and stateful web pages allowing the server to dictate and maintain the state of the page content.

FIG. 2B is a table illustrating how unintended page content is avoided regardless of whether caching is enabled. Table 500 includes three columns specifying, respectively, the address requested (C1), whether page content is loaded from local cache or host server (C2), and a description of the resulting page content (C3). Each row (R1 through R5) corresponds to a new request. FIGS. 2C through 2G illustrate various states of a web page displaying a car having selectable exterior and interior colors according to FIG. 2B. The web page has the same intended operation as discussed in connection with FIGS. 1C through 1G.

Referring to FIG. 2B, the address requested in table cell (R1, C1) is,

`http://www.mysite.com/PAGE1.HTML?&ID=123123678` (6).

The requested address includes a unique identifier appended to the URL in the form of a query string (*i.e.*, `?&ID=123123678`). Since the identifier is intended to be unique, there is no cached content associated with the URL/query string (6). Thus, an HTTP request
 5 specifying the URL/query string (6) is generated and transmitted over the network to the host server 210. The server 210 processes the request and returns the script file, `PAGE1.HTML`, to the browser 110. The server 210 may ignore the unique identifier in the query string, since its main function is to uniquely identify the URL/query string address preventing the browser 110 from loading content from cache. As shown in
 10 FIG. 2C, the page content resulting from the initial content request is a car 405 having a “black” exterior color and a “grey” interior color, as indicated by the car seat 410. The script file corresponding to this display, `PAGE1.HTML`, is stored in local cache 110 where it is available to be loaded when URL (6) is requested again. However, since the script file, `PAGE1.HTML`, is requested with a unique identifier that is different in each
 15 request, it is highly unlikely that the browser 110 will load the script file, `PAGE1.HTML`, from cache again.

In FIG. 2C, the user clicks on an exterior color palette 420 in the display with a pointing device, such as a mouse, to change the exterior color of the car to “red”. Referring back to FIG. 2B, the click event triggers an event handler, defined in
 20 `PAGE1.HTML`, that generates a new content request to display the car with a “red” exterior. In table cell (R2,C1), the requested address is,
`http://www.mysite.com/PAGE1.HTML?EXT_COLOR=”RED”&ID=19283192873` (7)
 The requested address includes an appended query string parameter specifying the “red” exterior color (*i.e.*, `?EXT_COLOR=”RED”`) as well as a unique identifier
 25 (*i.e.*, `&ID=19283192873`). Again, there is no cached content associated with the requested URL/query string address (7) due to the unique identifier appended to the query string. Thus, an HTTP request is generated and transmitted over the network to the host server 210 specifying the URL/query string (7). The server 210 responds with a modified script file, `PAGE1.HTML` for displaying the car 405 with “red” exterior and

“grey” interior colors, as shown in FIG. 2D. The “grey” interior color is derived from the previous state of the web page (*i.e.*, FIG. 2C), with the state being maintained at the server. The modified script file, PAGE1.HTML, is stored in the local cache where it is available to be loaded when URL/query string (7) is requested again. However, since

5 the script file, PAGE1.HTML, is requested with a unique identifier that is different in each request, it is highly unlikely that the browser 110 will load the script file, PAGE1.HTML, from cache again.

In FIG. 2D the user clicks on an interior color palette 430 in the display to change the interior color of the car to “black.” Referring back to FIG. 2B, the click

10 event triggers an event handler, defined in PAGE1.HTML, generating a new content request to display the car with a “black” interior. In table cell (R3,C1), the requested address is,

[http://www.mysite.com/PAGE1.HTML?INT_COLOR="](http://www.mysite.com/PAGE1.HTML?INT_COLOR=)BLACK"&ID=123612 (8)

The requested address includes an appended query string parameter specifying the

15 “black” interior color (*i.e.*, ?INT_COLOR="BLACK") as well as a unique identifier (*i.e.*, &ID=123612). Again, there is no cached content associated with the requested URL/query string address (8) due to the unique identifier appended to the query string. Thus, an HTTP request is generated and transmitted over the network to the host server 210 specifying the URL/query string (8). The server 210 responds with a modified

20 script file for displaying the car with “red” exterior and “black” interior colors, as shown in FIG. 2E. The “red” exterior color is derived from the previous state of the web page (*i.e.*, FIG. 2D). The modified script file, PAGE1.HTML, is again stored in the local cache 120 where it is available to be loaded when URL/query string (8) is requested again. However, since the script file, PAGE1.HTML, is requested with a unique

25 identifier that is different in each request, it is highly unlikely that the browser 110 will load the script file, PAGE1.HTML, from cache again.

Referring back to FIG. 2B, assume that the user closes the browser and relaunched a new session at a later time. From the previous user session, the script file, PAGE1.HTML, for the initial content of the web page was cached and associated with

the URL/query string (6) that is identified in table cell (R1, C1). However, when the user attempts to load the initial web page, the URL address is the same, but the address is appended with a different unique identifier. For example, in table cell (R4, C1), the requested address is

5 <http://www.mysite.com/PAGE1.HTML?&ID=8798456> (9)

Since the URL/query string address (9) is not identical to URL/query string address (6), an HTTP request is generated and transmitted over the network to the host server 210 specifying the URL/query string (9). The server 210 responds with a modified script file for displaying the default car with “black” exterior and “grey” interior colors, as shown in FIG. 2F. The script file, PAGE1.HTML, is again stored in the local cache 120, where it is available to be loaded when URL/query string (9) is requested again. However, since the script file, PAGE1.HTML, is requested with a unique identifier that is different in each request, it is highly unlikely that the browser 110 will load the script file, PAGE1.HTML, from cache again.

15 In FIG. 2F, the user clicks on an interior color palette 430 in the display to change the interior color of the car to “black”. Referring back to FIG. 2B, the click event triggers an event handler, defined in PAGE1.HTML, generating a new content request to display the car with a “black” interior. In table cell (R5,C1), the requested address is,

20 http://www.mysite.com/PAGE1.HTML?INT_COLOR=BLACK&ID=458 (10)

The requested address includes an appended query string parameter specifying the “black” interior color (*i.e.*, ?INT_COLOR=“BLACK”) as well as a unique identifier (*i.e.*, &ID=458). Again, there is no cached content associated with the requested URL/query string address (10) due to the unique identifier appended to the query string.

25 Thus, an HTTP request is generated and transmitted over the network to the host server 210 specifying the URL/query string (10). In contrast to FIG. 1B and 1G which loaded the page content from the cache, the server 210 responds with a modified script file for displaying the car with “BLACK” exterior and “black” interior colors, as shown in FIG. 2G. By appending the unique identifier to the requested address, the browser

110 is forced to transmit the content request to the server 210, because cached content is not associated with the unique address/identifier pair. Thus, the state of the web page is maintained as intended.

According to an embodiment of the invention, the following segment of

- 5 JavaScript code replaces a displayed web page with new page content from a web server regardless of whether the browser is configured to always load from the cache first.

```

function getWebSite( )      {
    parent.document.location.replace("http://www.mysite.com?&ID="
    +Math.random());
10 }

```

- Web browsers, such as Microsoft's Internet Explorer and Netscape's Navigator, also provide capabilities for navigating forwards and backwards among web pages already visited during a browser session. Typically, web browsers implement these capabilities through a navigational toolbar that includes elements, such as "Back" and
- 15 "Forward" action buttons. By clicking the "Back" or "Forward" button, the browser can request and load a web page previously visited during a browser session. The URL/query string for a previous web page is retrieved from a temporary history cache, which maintains an ordered list of URL/query strings used in previous requests for web pages. In the case, where a browser is configured to always attempt to load from cache
- 20 first, a "Back" or "Forward" navigational event will cause the browser to load page content from cache.

- For example, referring to FIG. 2B, assume that a user is currently viewing the car with "black" exterior and "black" interior colors, which resulted from the content request identified in table cell (R5, C1). If the user clicks the "Back" button in the
- 25 toolbar of the browser, the address identified in table cell (R4, C1), including the same unique identifier is requested again. In this case, there is cached content corresponding to the requested URL/query string. Thus, the browser loads content directly from the cache. This results in the browser displaying a car with "black" exterior and "grey"

interior colors, losing the changes made in the previous content request identified in row R4.

In certain web applications, a web designer may want changes in subsequent pages to affect the page content of earlier or later pages. Thus, if the user makes a
 5 change to one or more web pages that affect the content or have the ability to affect the content of earlier or later web pages, the browser must transmit the content request to the server rather than load prior page content from cache. In this way, changes made in subsequent pages can be reflected in related pages when a user navigates between pages.

The following JavaScript code segment in a script file, such as PAGE1.HTML,
 10 determines whether the current page is being loaded from cache and, if so, causes the browser to reload the page from the server:

```

originalReload = parent.step1_Reload;
if (originalReload != true)
    parent.step1_Reload = true;
15 else
    {
        parent.step1_Reload = false;
        document.location.replace("www.mysite.com/PAGE1.HTML?&ID=" +
            Math.random());
20     }

```

The boolean flag variable, step1_Reload, is maintained in a parent object (e.g., window or frame) of a child object (e.g., window or frame) that contains the script file, PAGE1.HTML. If the "parent.step1_Reload" flag is set to "false," the script is being loaded from the server and is allowed to continue. Conversely, if the
 25 "parent.step1_Reload" flag is set to "true," the script is being loaded from cache and is prevented from completing by submitting a new unique request for page content from the server. As illustrated in the code segment, a unique identifier is appended to the

URL/query string, in which the unique identifier is generated by the JavaScript method, Math.random().

When the script file is loaded from the server for the first time, the parent.step1_Reload flag is initially set to false. Thus, the script file is allowed to
5 continue loading from the server with the flag being changed to "true." When the page is requested again, such as in response to a navigational "Back" event, the page starts to load from cache, because there is cached content that corresponds to the same URL/query string, including the same unique identifier. As the browser executes the above code segment, the parent.step1_Reload flag is changed back to "false" and a new
10 request for the script file is submitted with a new unique identifier. The new script file from the server is allowed to load, since the 'parent.step1_Reload' flag is now set to "false."

The above described code segment is also triggered in response to another browser event, the "refresh" or "reload" of a page currently under view. In most
15 browsers, there is a "refresh" or "reload" button in the browser toolbar, which is used to update the contents of the page being viewed (e.g., static stock ticker web pages) or to reload a page that failed to load properly (e.g., certain graphics missing in display). When a "refresh" or "reload" is attempted, the browser requests the current URL/query string. When caching is enabled, the browser loads the page content directly from
20 cache.

However, by implementing a code segment, similar to the above described code segment, a "refresh" or "reload" event will trigger a new unique content request to the server every time as described above. This ensures that the most current page content is displayed to the user upon activating the "refresh" or "reload" action button.

25 FIG. 3 is a diagram illustrating how the display of unintended page content is avoided in response to redirection by a server according to one embodiment. A client computer 100 executing a web browsing application 110, which is capable of storing and loading content from a local cache 120. In this example, the local cache 120 stores a script file for displaying page content of a web page having an address,

`http://www.mysite.com/PAGE2.HTML` (11).

Assume that the browser 110 transmits a content request for PAGE1.HTML (reference 300) and the address includes a query string directing the web server to transmit the next page after PAGE1,

5 `http://www.mysite.com/PAGE1.HTML?ACTION=NEXTPAGE&ID=1111` (12)

In response to the request, the web server transmits a message redirecting the browser to PAGE2.HTML on the same server (reference 301). However, in order to prevent the browser from loading PAGE2.HTML from cache. The web server transmits the redirection message specifying the Internet address with a unique identifier, such as

10 `http://www.mysite.com/PAGE2.HTML?&ID=2222` (13)

When the browser checks its local cache for the URL/query string (13), there will be no content associated with the same unique address/identifier pair. Therefore, the browser 110 transmits a new content request (reference 302) to the server 210 requesting content for URL/query string (13). Thus, the browser 110 receives updated or stateful content

15 from the server 210 as intended (reference 303).

FIG. 4 is a flow diagram illustrating a process for displaying dynamic or stateful page content in a page-caching browser according to one embodiment. This process avoids displaying unintended page content.

At 600, the browser loads in a web page implemented according to an
20 embodiment of the invention.

At 610, while the web page is loading and executing, a global reload flag, such as `parent.step1_Reload`, is set to false. This flag indicates that the page content is being loaded from the server and is current.

At 620, while the web page is loading and executing, the global reload flag is
25 tested to identify whether the page content is being loaded from cache or from the server. If the reload flag is set to false, then the page is being loaded from the server and the process proceeds to 630. Otherwise, if the reload flag is set to true, the page is being loaded from cache and the process proceeds to 690 to force a reload of the page content from the server.

At 630, since it is determined that the page is being loaded from the server, the page content is current and no reload of the page is required. The global reload flag, however, is set to true to prevent this page from being loaded from cache when requested again.

5 At 640, the page continues to load and execute.

At 650, an event is triggered. The response to the event depends on the type of event, such as a reload event (660), back/forward/history navigational event(670), or a page event (680).

At 665, if the event is a reload event, the browser requests the address
10 (*i.e.*, URL/query string) of the current web page.

At 667, since the cache maintains content for the requested URL/query string, including the unique identifier, the browser loads and execute the corresponding content from the cache returning to 620.

At 620, the global reload flag is checked to determine whether the content is
15 being loaded from cache. Since the reload flag was set to true at 630, the process proceeds to 690 to request updated page content from the server.

At 690, the global reload flag is set to false indicating that the page is being loaded from the server.

At 695, the browser requests the current URL/query string address with a unique
20 identifier appended to the address for updated page content.

At 697, since there is no cached content corresponding to the address and unique identifier, the page content is loaded from the server.

Again, as the new page content is loaded and executed by the browser in response to the request of 695, the reload flag is checked at 620. Since the reload flag is
25 currently set to false, the process is able to proceed to 630 and 640, to display the updated page content from the server.

Referring back to 660, if the event is not a reload event, then it may be a back/forward/history event (670). If yes, the process proceeds to 675.

At 675, the browser requests the URL/query string, including the same unique identifier, of a previously visited page.

At 677, since the re-requested URL/query string is associated with page content in the cache, the browser initially loads the cached content, but will be forced to request
5 new page content as illustrated in the reload case at 620/690/695/697.

If the event is neither a reload or historical navigation event, the event is typically a user-defined page event (680).

At 683, the global reload flag is set to false indicating that new page content from the server is being requested.

10 At 685, the new URL/query string address with a new unique identifier appended is requested.

At 687, since there is no cached content associated with the unique address, the page content is loaded from the server.

At 620, during the loading and executing of the page content, the reload flag is
15 checked. Since the reload flag was set to false in 683, the page content continues to load executing at 630 and 640.

In sum, embodiments of the invention include a system and method for displaying dynamic and stateful page content in a page-caching browser by allowing a browser to load page content from a host server regardless of whether caching is
20 enabled. Therefore, the display of unintended page content is avoided facilitating the use of stateful web pages regardless of whether a browser is configured to always attempt to load content from its local cache. Embodiments of the invention can be implemented even if caching is not enabled. Thus, separate scripts do not need to be maintained for caching systems and non-caching systems.

25 Those of ordinary skill in the art realize that methods involved in a system and method for displaying dynamic page content in a page-caching browser may be embodied in a computer program product that includes a computer-usable medium. For example, such a computer usable medium can include a readable memory device, such as a hard drive device, a CD-ROM, a DVD-ROM, a computer diskette or solid-state

memory components (ROM, RAM), having computer readable program code segments stored thereon. The computer readable medium can also include a communications or transmission medium, such as a bus or a communications link, either optical, wired, or wireless, having program code segments carried thereon as digital or analog data

5 signals.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197